**Цель работы:** изучение машинного языка.

Список заданий:

1. Для изучения структуры двух команд (пересылки MOV и сложения ADD), в которых используются непосредственные данные необходимо в левом верхнем окне отладчика написать указанные команды, в строках, нумеруемых последовательно. В одной строке напишите команду для пересылки значения 4629h в регистр AX, а в следующей строке напишите команду сложения числа 036Ah с содержимым регистра AX.
2. Для понимания правил, по которым компьютер обеспечивает доступ к данным посредством регистров SI, DI, DS предлагается пошагово выполнить следующую программу. В верхнем левом окне отладчика напишите в строках, нумеруемых последовательно, команды пересылки значений 3, 8, 2100 соответственно в регистры SI, DI, DS. Задайте в памяти процессора (нижнее левое окно) по эффективному (смещенному) адресу 3 значение 7Ch. Далее напишите команды пересылки MOV AL,[SI] и MOV [DI],AL.
3. Для изучения способа кодирования символов представьте в памяти процессора (нижнее левое окно) строку символов, соответствующих Вашему имени и году рождения, используя 8-разрядные коды ASCII.
4. Для изучения машинного (двоичного) представления целой и дробной частей числа в формате с фиксированной точкой (запятой) напишите в верхнем левом окне отладчика фрагмент программы, заданный ниже. Каждое непосредственное данное, используемое во фрагменте программы, представляется в виде двух байтов. Для представления дробной части используется один байт и для представления целой части также используется один байт.

MOV AX,1080; AX = 1080h = 16.5d

MOV BX,1240; BX = 1240h = 18.25d

ADD AX,BX; AX = 22C0h = 34.75d

Выполните в пошаговом режиме указанный фрагмент программы, анализируя значения разрядов регистра флагов. Найдите такие значения складываемых чисел, которые обеспечивают изменения значений флагов

1. Для изучения машинного (двоичного) представления коротких вещественных чисел, задаваемых в формате с плавающей точкой, напишите и выполните фрагмент программы, заданный ниже.

FINIT ; инициализация FPU (например, 8087) TOP = 0.

FLDZ ; поместить в стек +0.0 (TOP – 1 и поместить в ST(0) константу).

FLD1 ; поместить в стек 1.0.

FADD ST(0),ST(1) ; результат сложения чисел 0.0 и 1.0 размещен в ST(0).

FST dword ptr[0] ; копирование ST(0), TOP не изменяется.

FLD1

FADD ST,ST(1) ; результат сложения чисел размещен в ST(0).

FST dword ptr[4]

FLD1

FADD ST,ST(1) ; результат сложения чисел размещен в ST(0).

FST dword ptr[8]

FLD1

FADD ST,ST(1) ; результат сложения чисел размещен в ST(0).

FST dword ptr[0c]

FDIVR ST,ST(2) ; вещественной число ½ размещено в ST(0).

FST dword ptr[10]

1. Задайте отображение сегмента данных ds:0000, ds:0004 и так далее. Приведите формулу вычисления q и p двоичного представления вычисленных чисел. Приведите формулу и двоичное представление максимального и минимального положительных денормализованных и нормализованных чисел.

Задание 1

Тест программы:

Mov ax, 4629h; ax = 4629h

Add ax, 036Ah; ax = ax + 036Ah

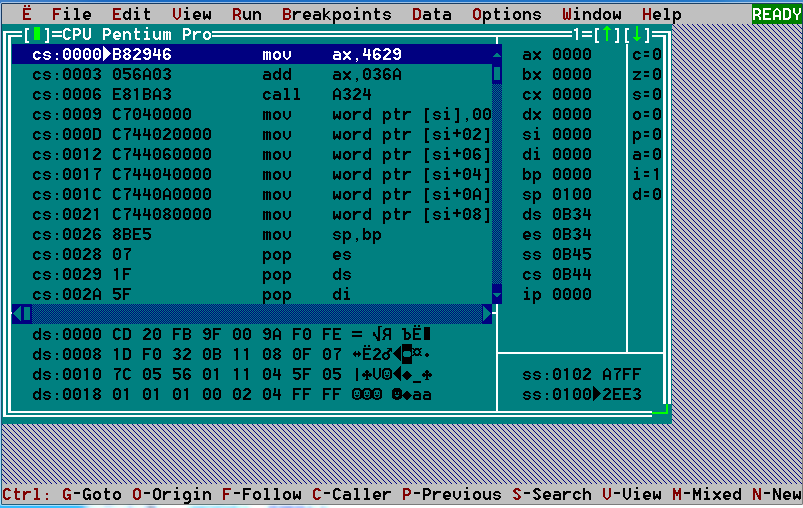


Рис. 1.1 – Программа до отладки

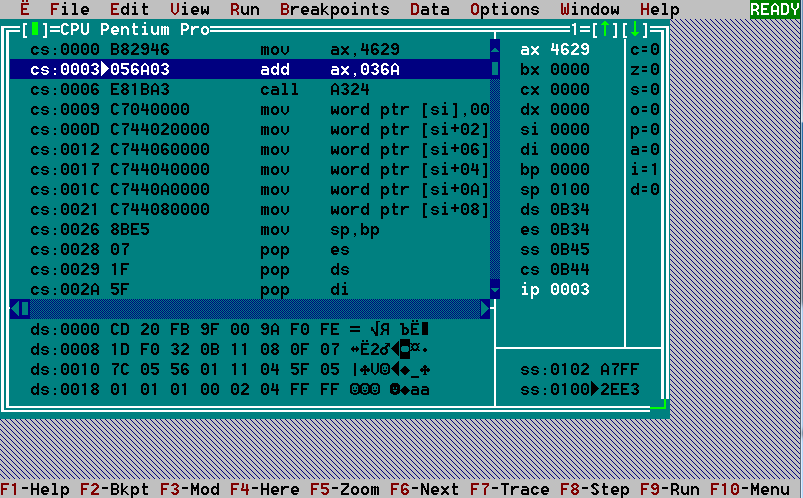


Рис. 1.2 – Выполнена Mov ax, 4629h; ax = 4629h

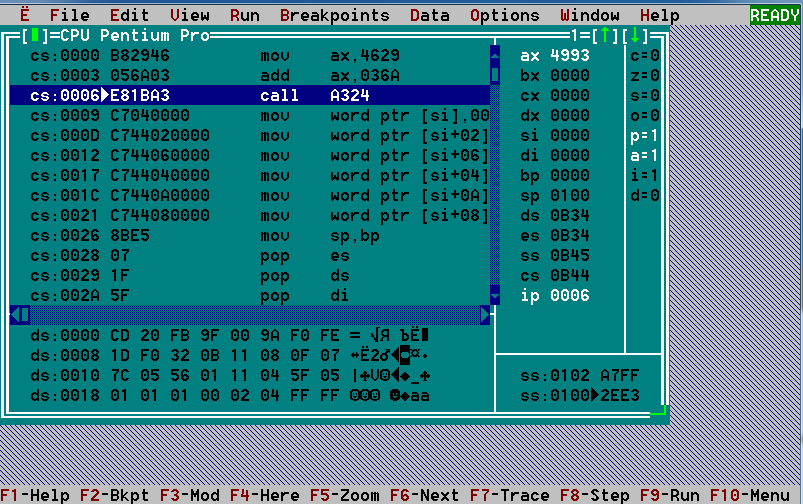


Рис. 1.3 – Программа после отладки

Результат работы программы:

1. В регистр ax записывается значение 4629h, h – шестнадцатеричная система счисления.
2. В регистр ax суммируется число 036Ah в шестнадцатеричной системе вычисления и в результате в ax записано число 4993 в шестнадцатеричной системе счисления

Задание 2

Тест программы:

Mov si, 0003; si = 3

Mov di, 0008; di = 8

Mov ax, 2100; ax = 2100

Mov ds, ax; ds = ax

Mov byte ptr [si], 7Ch; [si] = 7Ch

Mov al, [si]; al = [si]

Mov [di], al; [di] = al

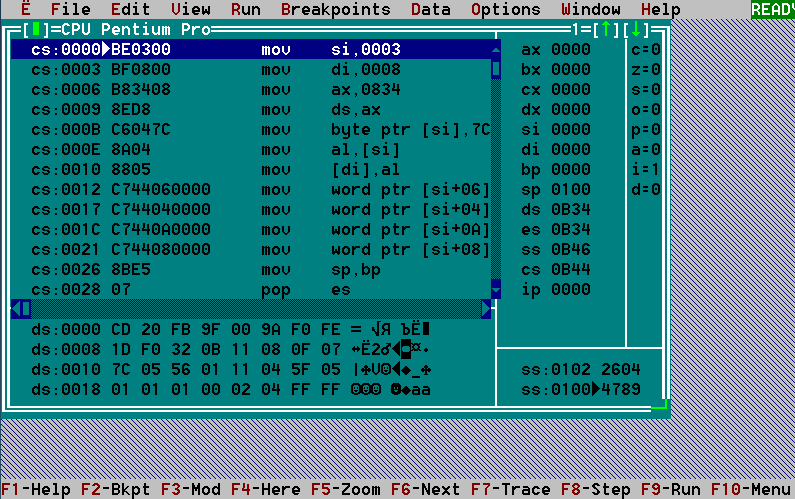


Рис. 2.1 – Программа до отладки

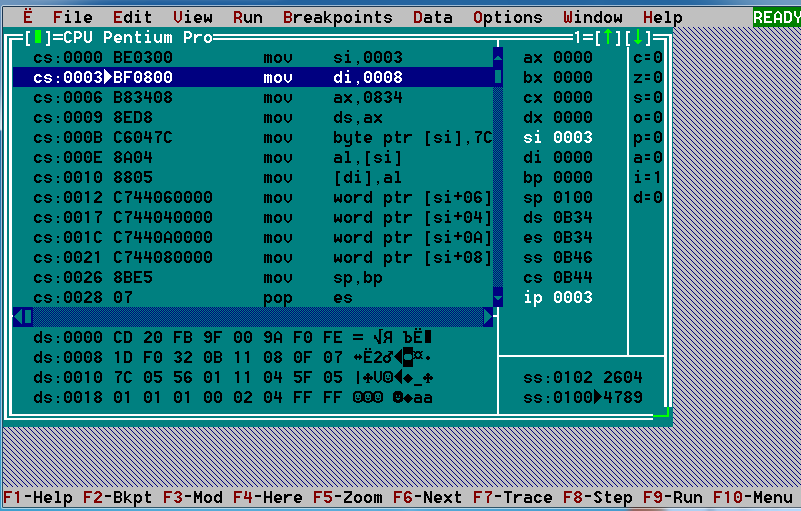


Рис. 2.2 – si = 3

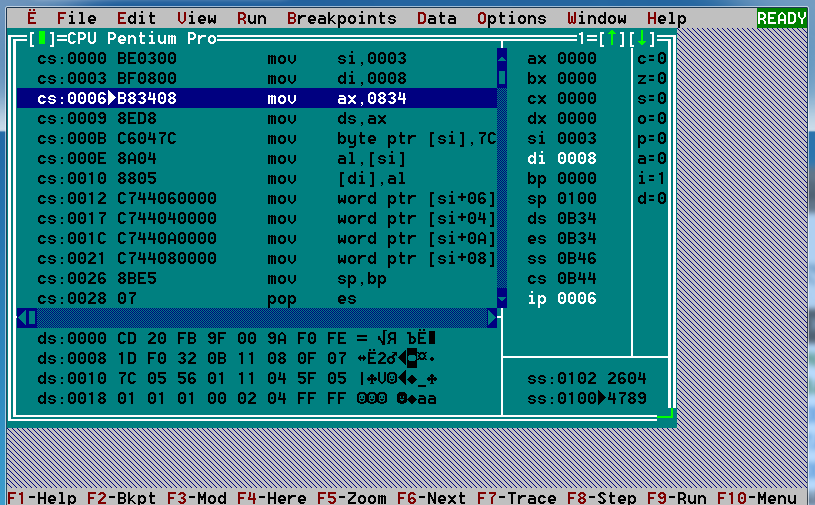


Рис. 2.3 – di = 8

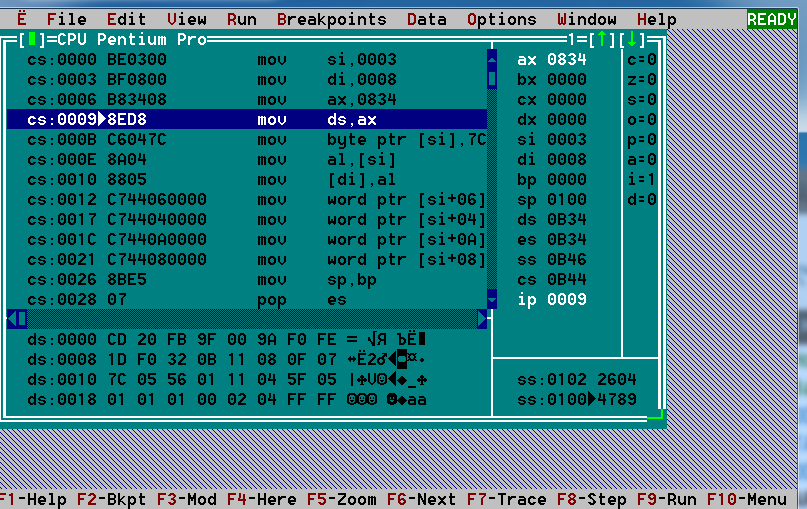


Рис. 2.4 – ax = 2100

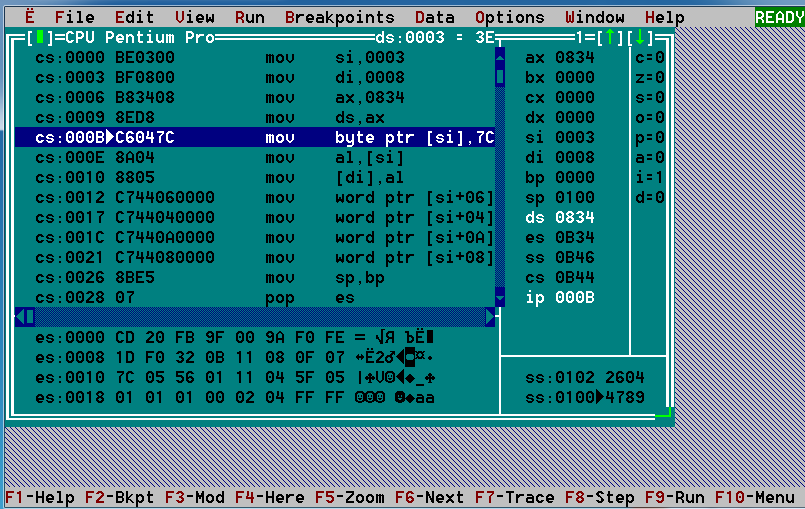


Рис. 2.5 ds = 2100

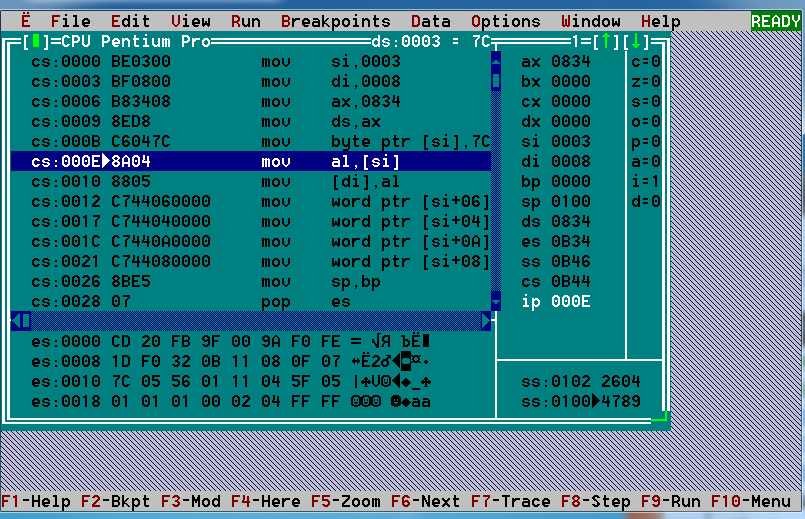


Рис. 2.6 0x0003=7C

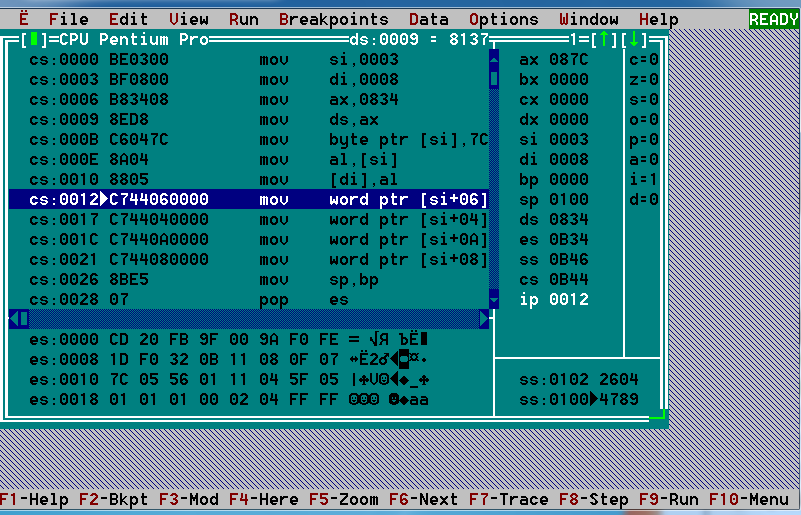


Рис. 2.7 – Программа после отладки

Результат программы:

1. Записываем в регистр si (индекс источника) значение равное 3.
2. Записываем в регистр di (индекс приемника) значение равное 8.
3. Через регистр ax записываем данные в регистр ds (сегментный регистр данных).
4. Используя регистр al промежуточного хранения, копируем значение ячейки по адресу si в ячейку по адресу di.

Задание 3

В сегмент данных, начиная с адреса ES:0000 была записана следующая последовательность ASCII символов:

56 69 63 74 6F 72 00 00

31 39 39 37 00 00 00 00

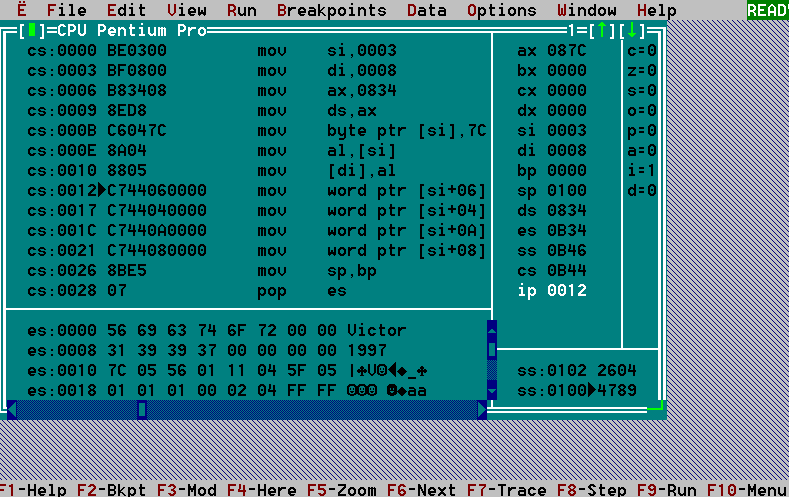


Рис. 3.1 – Результат работы

В итоге, с помощью записи кодов ASCII в память процессора, мы можем наблюдать, что они преобразовались и отобразились на экране:

Victor

1997

Задание 4

Текст программы:

Mov ax,1080h; ax = 1080h = 16.5d

Mov bx,1240h; bx = 1240h = 18.25d

Add ax,bx; ax = ax + bx = 22C0h = 34.75d

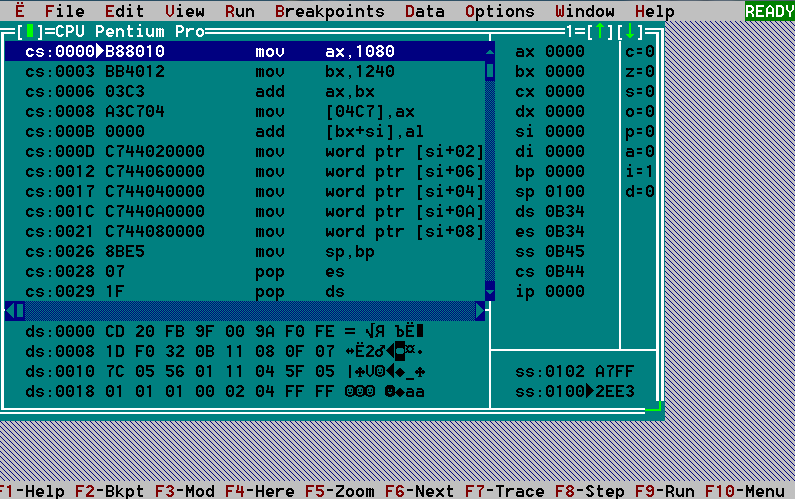
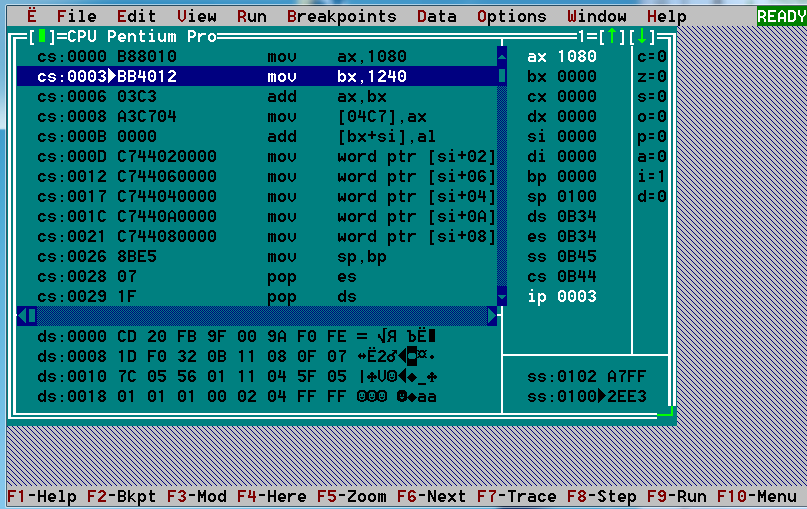
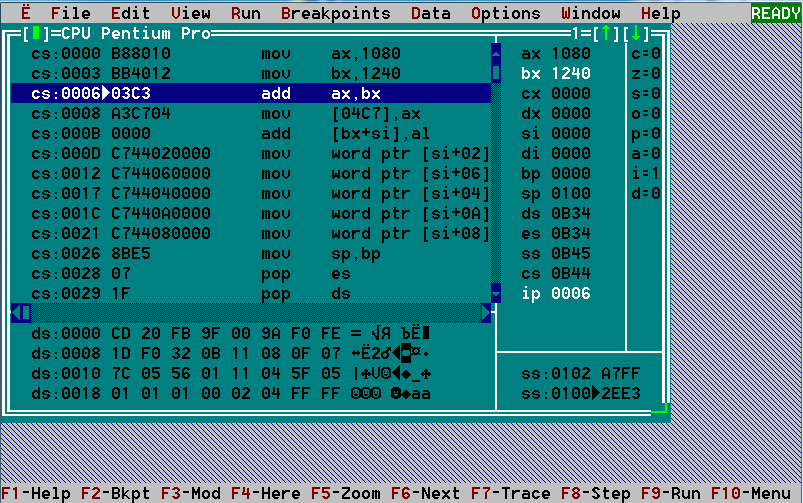


Рис. 4.1 – Программа до отладки





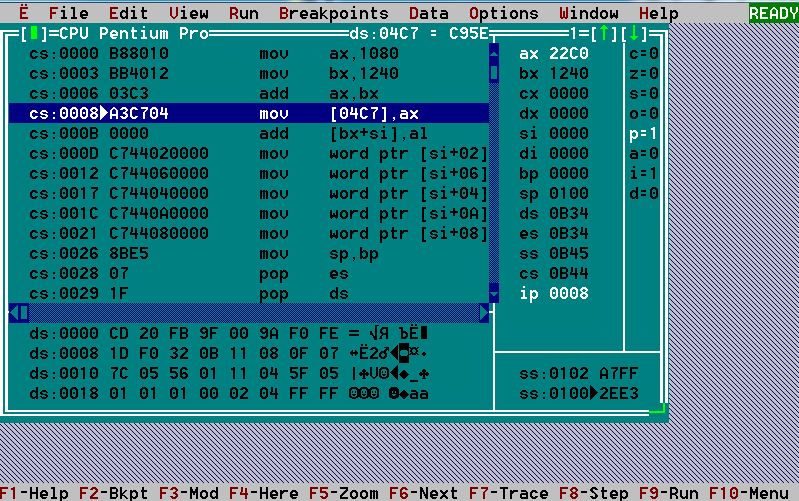


Рис. 4.2 – Программа после отладки

Результатом работы программы является сумма двух дробных числа, записанная в регистр ax.

Так как в регистр ax, размером 2 байта, может поместиться число до 65535 (2^16 -1), то, чтобы значения флагов поменялись нужно подобрать такие числа, при сложении которых включится флаг переполнения Carry Flag. Я подобрала числа 7FFFC0=32767.75 и 7FFFC0=32767.75.

Результат сложения этих чисел приведен на рисунке 4.3.

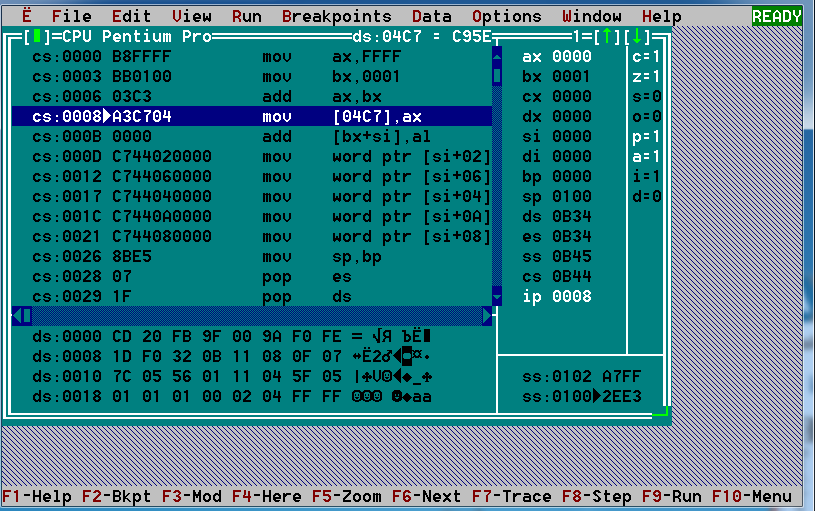


Рис. 4.3 – Программа после отладки

Результат программы:

В связи с переполнением, Carry Flag поменял свое значение на 1, можем наблюдать в правом верхнем углу.

**Задание 5**

Текст программы:

FINIT; инициализация FPU (например, 8087) TOP = 0.

FLDZ; поместить в стек +0.0 (TOP – 1 и поместить в ST(0) константу).

FLD1; поместить в стек 1.0.

FADD ST(0),ST(1); результат сложения чисел 0.0 и 1.0 размещен в ST(0).

FST dword ptr[0] ; копирование ST(0), TOP не изменяется.

FLD1; поместить в стек 1.0.

FADD ST,ST(1); результат сложения чисел размещен в ST(0).

FST dword ptr[4]

FLD1; поместить в стек 1.0.

FADD ST,ST(1); результат сложения чисел размещен в ST(0).

FST dword ptr[8]

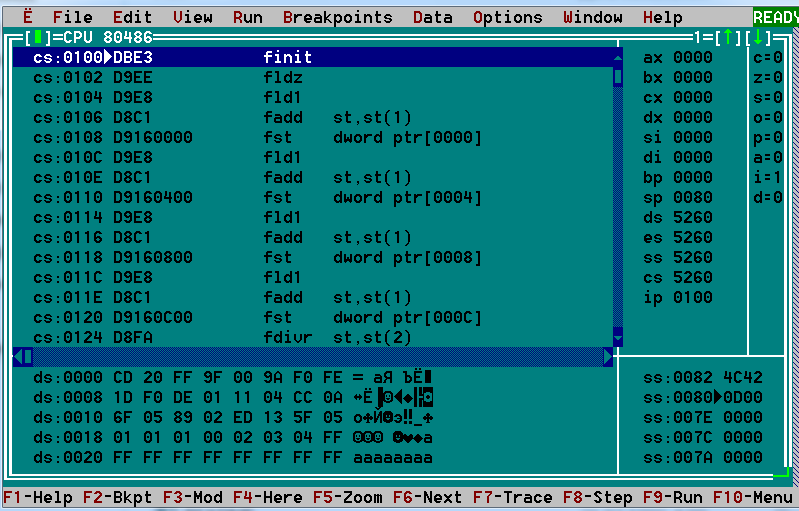
FLD1; поместить в стек 1.0.

FADD ST,ST(1); результат сложения чисел размещен в ST(0).

FST dword ptr[0c]

FDIVR ST,ST(2) ; вещественной число ½ размещено в ST(0).

FST dword ptr[10]



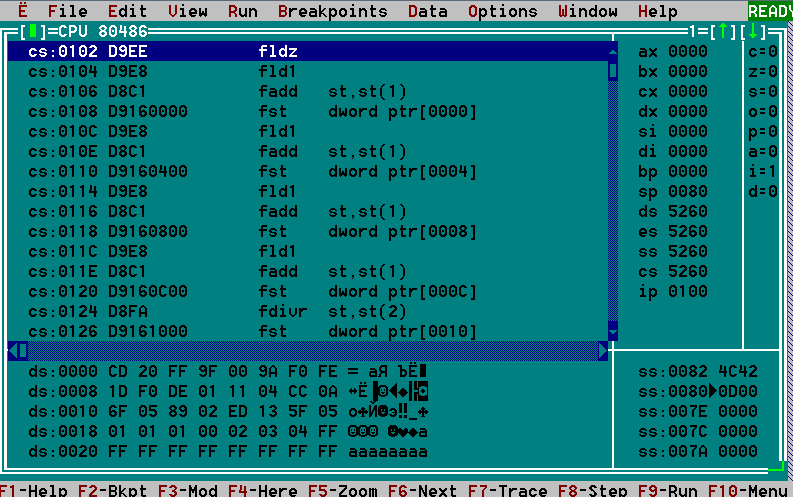


Рис. 5.1 и 5.2 – Программа до отладки

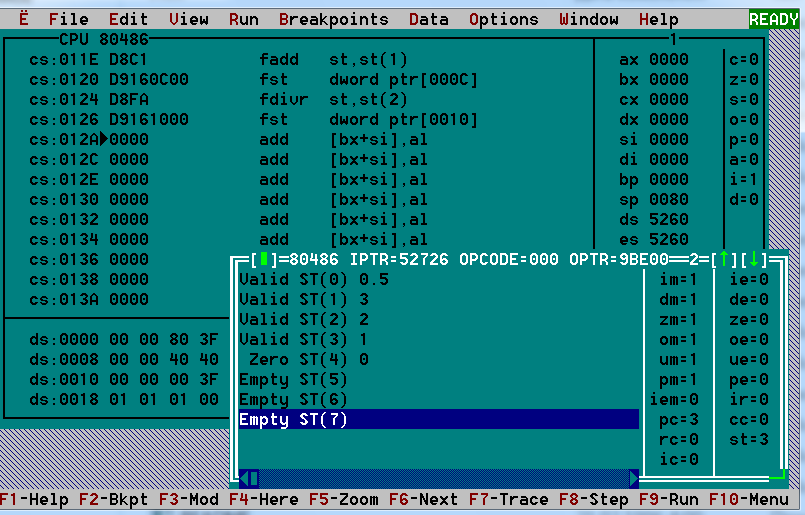


Рис. 5.3 – Программа после отладки

Результат работы программы:

1. На первом шаге инициализируем процессор
2. Затем командой FLDZ загружаем данные, т. е. 0, в вершину стека сопроцессора
3. После, командой FLD1 загружаем данные, т. е. 1, в вершину стека сопроцессора
4. Складываем значения из предыдущих двух пунктов и загружаем в вершину стека
5. Проделываем данные действия несколько раз
6. Чтобы загрузить вещественное число, используем FDIVR и размещаем его в вершине стека

Наглядный результат выполнения данной последовательности действий показан на рис. 5.4

Числа с плавающей запятой имеют размер 4 байта и следующий формат:

где q - целое, выражающее порядок числа X; m - мантисса числа X.

Подобные числа могут быть представлены в нормализованном и денормализованном видах. Отличием является строго заданное значение неявного бита мантиссы: у нормализованных - 1, у денормализованных - 0.

Нормализованные числа имеют следующий вид:

Денормализованные:

Где s - знаковый бит, М - мантисса, E – порядок, B – основание (в нашем случае это 2)

Нормализованные и денормализованные числа имеют различные диапазоны значений.

Для денормализованных чисел:

Минимальное возможное положительное значение:

Максимально возможное положительное значение:

Для нормализованных чисел:

Минимальное возможное положительное значение:

Максимально возможное положительное значение:

Учитывая то, что числа хранятся в нормализованном виде, рассмотрим результат, полученный в ходе выполнения программы:

Вывод:

В ходе лабораторной был изучен машинный язык – ассемблер.

Получены следующие навыки:

* Структурированное решение задач
* Использование операторов языка
* Формирование тестовых ситуаций
* Осуществление процесса отладки
* Документирование результатов работы
* Использование Turbo Debugger